Matthew Kemenosh, Connor Hsuan, Miguel Rodriguez, and Deep Shah

CPE 462: Introduction to Image Processing and Coding

5/15/25

Professor Man

"We pledge our honor that we have abided by the Stevens Honor System." ~ MK, CH, MR, DS

Final Project Report: The Image Processing ToolBox

Background/Inspiration

We determined our project: The Image Processing ToolBox in a fascinating manner. Originally, the group was interested in doing image enhancement and restoration. Instead, we came up with a better idea that heavily focused on image enhancement and general image techniques. We found that we want to focus on this topic as Deep Shah was researching and exploring different techniques to use for image enhancement. Additionally, he wanted to pursue and proposed an idea where he wanted to execute a project that is both impactful and significant in helping others out. He didn't just want to show the input and output for a specific image that we have. He instead wanted to pursue a project where we created a small tool kit for users to apply image enhancement and transformation techniques for each user's own image. I brought up the idea to my teammates and they were all fine with it, so we took the initiative to design: The Image Processing Toolbox.

Introduction

The Image Processing Toolbox is an interactive Python based tool that allows users to upload an image and apply one of ten selectable image transformations. The image transformation options are image resizing, image rotation, image translation, image shearing, image normalization, edge detection of an image, image blur, noise removal, brightness and

contrast adjustment, and color enhancement. Using the OpenCV and Matplotlib libraries, the program enables efficient visual manipulation of images for both enhancement and analytical purposes. The system uses a menu-driven interface that allows the user to upload an image and choose a transformation technique from the listed options. Additionally, a key feature that we have is that the images that are uploaded by the user can be from a variety of different file types. This is important as individuals have their images saved as different file types, so this would be significant and applicable for any user to use. The file types that can be uploaded to our tool kit for images are joint photographic experts group (JPEG/JPG) and portable network graphics (PNG).

Methods

Each function within the toolbox was developed using OpenCV to apply a specific image processing technique. Users are prompted to upload an image and input parameters specific to the selected transformation. Every image is read and saved using cv2.imread(). Image information is gathered using cv2.cvtColor(). Matplotlib is used to display the original and transformed images side by side for easy visual comparison. Examples of the techniques can be found in the Finding/Output section. Here is how each processing technique functions:

Image Resizing:

This option prompts the user for their desired width and height for the image. It utilizes cv2.resize to resize the image accordingly, changing the dimensions of the image based on the user's specified width and height. Then it displays the original and resized images side by side.

Image Rotation:

This option requests a rotation angle from the user. It retrieves basic measurements of the image and uses OpenCV's rotation matrix to rotate the image around its center, specifically

employing cv2.getRotationMatrix2D() and cv2.warpAffine() to rotate the image around its center point at the user defined angle. Then it displays the original and rotated images.

Image Translation:

This option takes user input for X and Y translation values. It then shifts the image along the x and y axes using cv2.warpAffine(), an affine transformation matrix. Then it displays both the original and translated images.

Image Shearing:

This option asks the user for shear values along the X and Y axes. Constructs a shear matrix and applies affine transformation. Applies a shear transformation using a manually defined affine matrix cv2.warpAffine() with shear factors in X and Y directions. Then it displays the sheared result alongside the original.

Image Normalization:

This option splits the image into R, G, B channels using cv2.split(). Each color channel of the image is then normalized to scale pixel values between 0 and 1 using cv2.normalize(). It then merges the image with cv2.merge() and displays the normalized image.

Edge Detection of an Image:

This option applies the Canny edge detection algorithm using cv2.Canny() with predefined thresholds to detect sharp changes in image intensity. It then outputs an edge-detected image beside the original.

Image Blur:

This option requests a blur kernel size amount (odd integer) from the user. It applies a Gaussian blur to the image using cv2.GaussianBlur() based on the user specified kernel size and collects its information. It then outputs the original and blurred images.

Noise Removal:

This option applies a median blur filter to the image with kernel size 11. It reduces image noise with a median blur using cv2.medianBlur(), which is commonly used to preserve edges while smoothing. Then it displays the original and noise-reduced images.

Brightness and Contrast Adjustment:

This option receives user inputs for image contrast (alpha) and brightness (beta) values. It modifies the image brightness and contrast using cv2.convertScaleAbs() with the user defined alpha and beta values. It then outputs the original and adjusted images.

Color Enhancement:

This option converts the image to HSV color space to enhance its colors. Each channel is adjusted, modifying hue, saturation, and value for enhancement. It then converts back to BGR and displays the resulting enhanced image and the original.

User Interface:

Upon running the program, users are presented with a numbered menu of the ten image transformation options and an option to exit. Each function begins with a file upload dialog, followed by prompts for necessary parameters, then displays the original and processed image for visual comparison.

Finding/Output

Our Image Processing Tool is able to receive a large variety of formats and alter the image depending on the request. In the following examples, I used a jpg file and altered it in several ways.

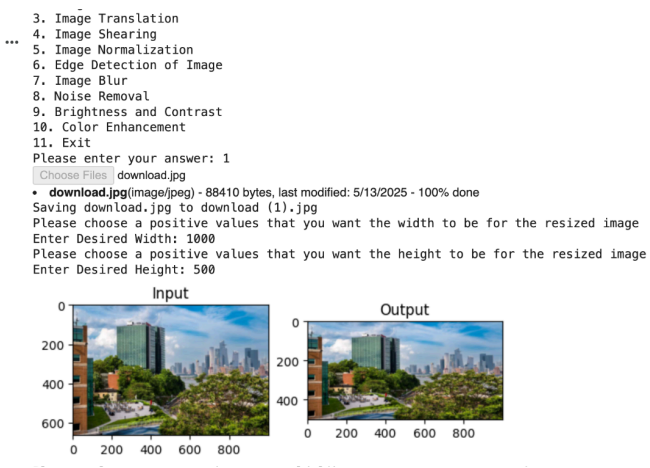

Figure 1

In this first example (*Figure 1*), I used the Image Resizing function to resize my original image into an image that had a width of 1000 pixels and a height of 500 pixels. The resized version of the image can be seen on the right with an appropriate pixel scale.

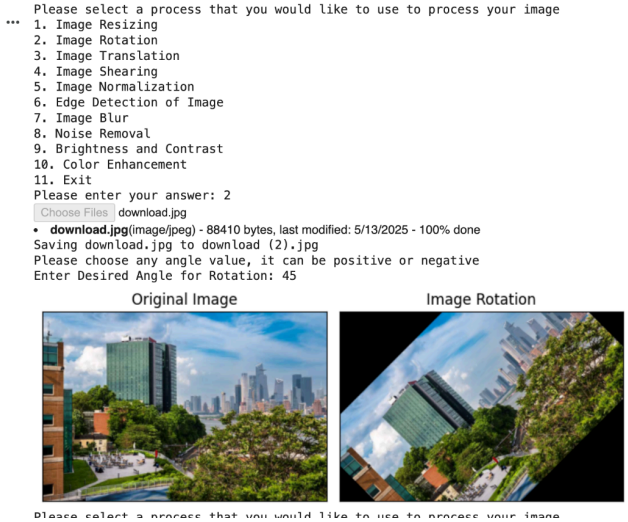

Figure 2

In this next example (*Figure 2*), I used the Image Rotation function to rotate my original image 45 degrees. If I had inputted a negative number rather than a positive one, the image would have been rotated clockwise instead of counterclockwise.

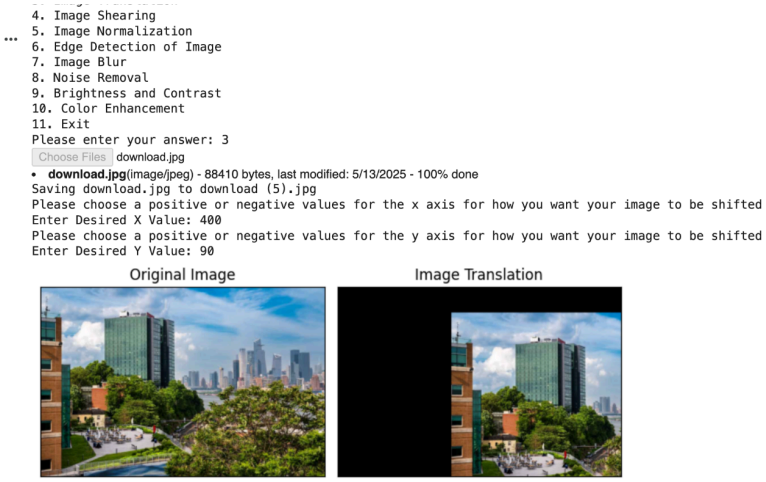

Figure 3

In the third example (*Figure 3*), I used the Image Translation function to move the image 400 pixels to the left and 90 pixels downward. Using negative values for your inputs would have resulted in the image being shifted in the opposite direction.

Figure 4

In the fourth example (*Figure 4)*, I used the Image Shearing function to shift the image 3 units to the right and 2 units down. An image shearing value that was too great would shift the image too greatly, causing the output image to look like a black screen. Negative numbers would also shear the image in the opposite direction.

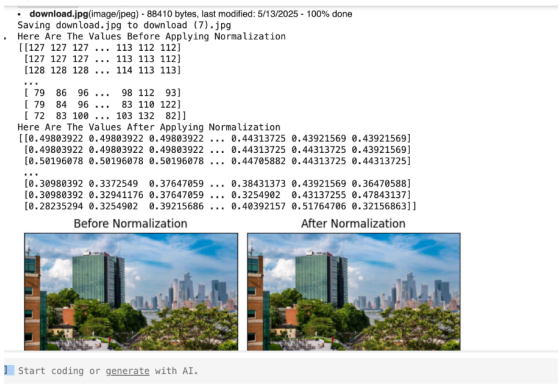

Figure 5

For the fifth example (*Figure 5)*, I used the Image Normalization function to apply a normalization to my image. The values before and after the normalization are shown at the top and the before and after image is shown on the bottom.

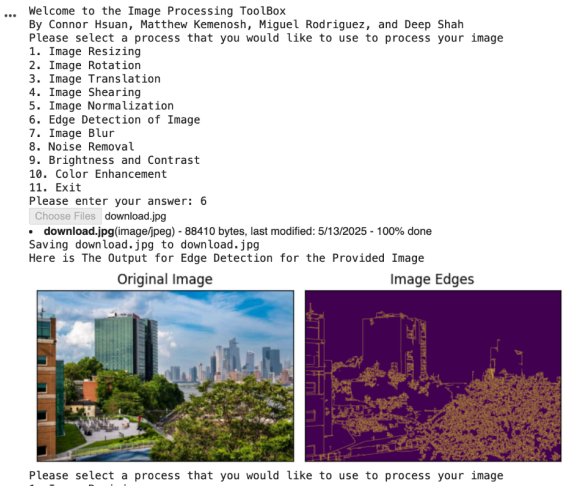

Figure 6

For the sixth example (*Figure 6*), I used the Edge Detection of Image function to output the edges in an image. The original image is on the right and the output is on the left. The edges of the image are seen in yellow on a purple background.

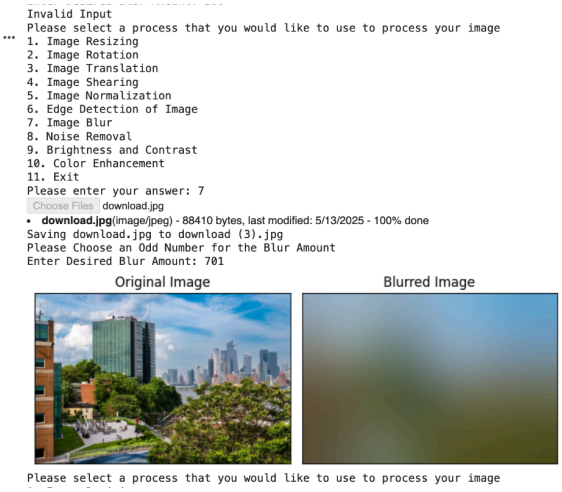

Figure 7

In the seventh example (*Figure 7*), I used the Image Blur function to blur the original image by a factor of 701. The original image can be seen on the left and the new blurred image can be seen on the right.

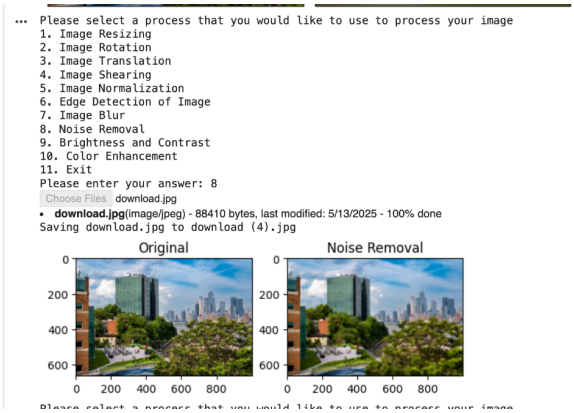

Figure 8

In the eighth example (*Figure 8*), I used the Noise Removal function to output a version of the image that had its noise removed. The original image is to the left and the noise removed version is to the right.

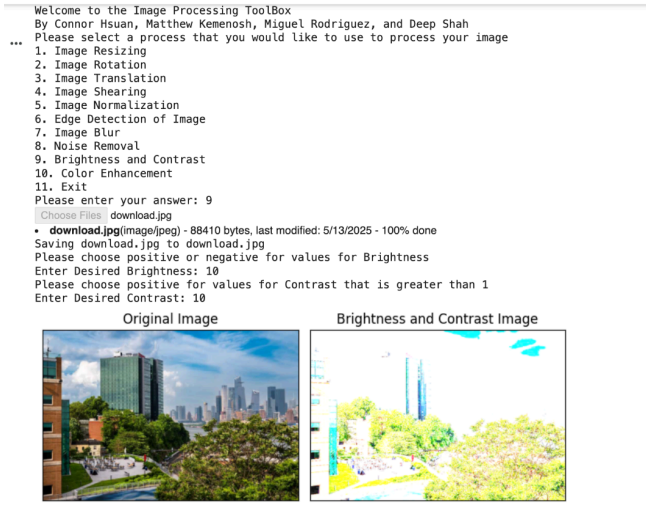

Figure 9

In the ninth example, I used the Brightness and Contrast function on my image to increase the brightness. My original image is on the left and the new image is on the right. The image has had its brightness increased by a factor of 10 and the contrast also increased by 10. Inputting a negative value for brightness would still brighten the image as if that number were positive.

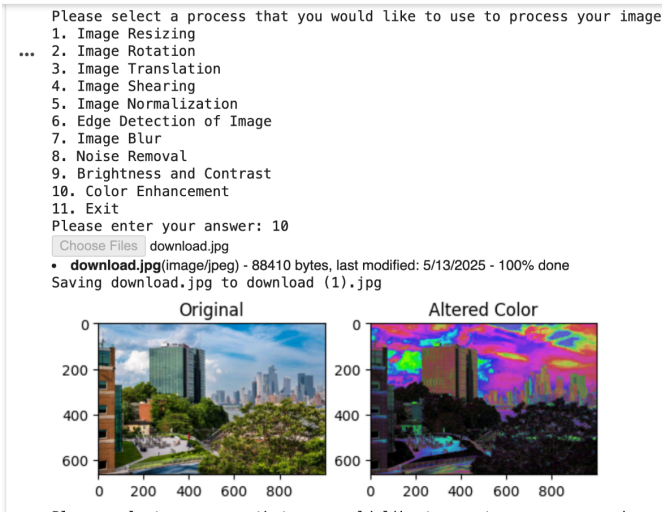

Figure 10

For the tenth example (*Figure 10*), I used the Color Enhancement feature to change the colors of my original image. The original image can be seen on the left and the new image with the altered colors can be seen on the right.

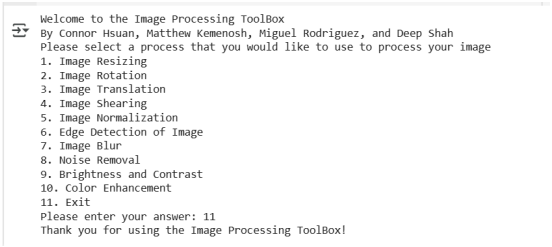

Figure 11

The last function (*Figure 11*), will simply exit the program. The program will display an exit message and then close.

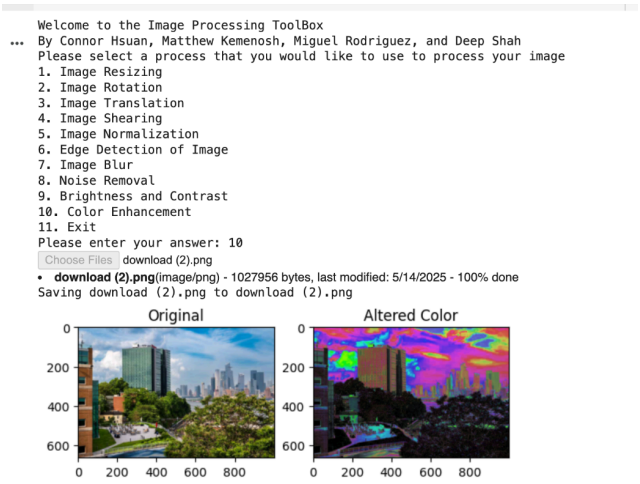

Figure 12

All of the functions work with several image format types. In this example (*Figure 12*) I used the Color Enhancement feature to alter a png of the same image as I used for the other example. The function works exactly as it did in the tenth example and the output is exactly the same.

Conclusion

Our Image Processing Toolbox is an interactive Python program that allows users to upload images of various types and alter them in 10 different ways. The list of functions is as follows: Image Resizing, Image Rotation, Image Transition, Image Shearing, Image Normalization, Edge Detection of an Image, Image Blur, Noise Removal, Brightness Contrast,

and Color Enhancement. Our program uses OpenCV and Matplotlib libraries to manipulate the images to the desired effect. Our menu system also provides a fast and efficient way to edit images and get the desired output. Our program employs a variety of methods to transform images in 10 different ways so that it would be fast and simple for the user.

Contributions Among Team Members:

Matthew helped in writing up the Introduction and Methods sections of the report and making edits in the other sections, as well as helping with some code bug fixes and testing.

Miguel helped construct the individual functions to physically change the images depending on the user's input into the program. Mainly worked on bug fixes, testing and cleaning up the code. Also helped write certain sections of the report.

For the project, the contributions that Deep made were that he wrote the background section for this report. In terms of the code, he constructed certain functions that were for image transformations and helped in making it accessible for user's input of their own images. Lastly, he cleaned up the code and tested it.

Connor helped with additional testing of the code and was responsible for writing the Finding/Output and Conclusion sections.